LISTING OF CLAIMS

The listing of claims provided below replaces all prior versions and listing of claims in the application. Please amend the claims as follows:

1. (Currently Amended) A method for ~~loop optimization within a~~ executing a computer program using a dynamic compiler system, the method comprising:

receiving a byte code of the computer program;

determining if native code is available that corresponds to the byte code;

executing the native code when the native code is available;

when the native code is not available, incrementing a counter;

when the counter is less than a threshold value, ~~executing interpreted byte codes of a computer program having a loop structure, wherein the loop structure includes a loop exit test to be performed during each loop iteration and counting a number of times each of the interpreted byte codes is executed~~ interpreting the byte code; and

when the counter is greater than the threshold value, compiling ~~the loop structure during the execution of the computer program~~ a subroutine containing the byte code~~; and~~

~~creating an unrolled loop structure during the compiling operation, the creating of the unrolled loop structure being initiated by interpreted byte codes that are often used, as identified by the number of times each of the interpreted byte codes is executed, and wherein the unrolled loop structure includes a plurality of loop bodies based on the loop structure~~.

2. (Currently Amended) A method as recited in claim 1, wherein the subroutine has a loop structure, wherein the loop structure includes a loop exit test to be performed during each loop iteration, the compiling comprising creating an unrolled loop structure, the unrolled loop structure includes a plurality of loop bodies based on the loop structure ~~the unrolled loop structure includes the loop exit test~~.

3. (Currently Amended) A method as recited in claim 2, wherein the unrolled loop structure includes a loop exit test, the loop exit test is performed once for each iteration of the plurality of loop bodies.

4. (Currently Amended) A method as recited in claim [[1]] 2, further comprising ~~the operation of~~ building a loop tree based on loops included in the computer program.

5. (Original) A method as recited in claim 4, wherein nested loops are represented in the loop tree as child nodes.

6. (Canceled)

7. (Currently amended) A method as recited in claim [[1]] 2, further including ~~the operation of~~ performing loop clean up.

8. (Original) A method as recited in claim 7, wherein the loop clean-up includes optimizing multiple fall-in loop structures.

9. (Original) A method as recited in claim 7, wherein the loop clean-up includes optimizing nested loop structures having invariant operations.

10. (Currently amended) A dynamic compiling system, comprising:

an interpreter ~~capable of interpreting~~ that interprets byte codes of a computer program ~~during execution of the computer program~~, the interpreter tracking a number of times a particular byte code is interpreted and requests ~~being further capable of requesting~~ that [[a]] the particular byte code be compiled~~, wherein the particular byte code is compiled based on a count of the~~ when the number of times the particular byte code is ~~executed~~ interpreted exceeds a threshold; and

a compiler ~~capable of compiling~~ that compiles the byte codes in response to the requests from ~~as requested by~~ the interpreter~~, wherein the compiler is further capable of creating an unrolled loop structure when compiling an original loop structure of the computer program, wherein the unrolled loop structure includes a plurality of loop bodies based on the original loop structure~~.

11. (Currently Amended) A dynamic compiling system as recited in claim 10, wherein the compiler is further capable of creating an unrolled loop structure when compiling an original loop structure of the computer program, the unrolled loop structure including a plurality of loop bodies based on the original loop structure ~~the unrolled loop structure includes the loop exit test~~.

12. (Currently Amended) A dynamic compiling system as recited in claim 11, wherein the unrolled loop structure includes the loop exit test, the loop exit test [[is]] being performed once for each iteration of the plurality of loop bodies.

13. (Currently Amended) A dynamic compiling system as recited in claim [[10]] 11, wherein the compiler is further capable of building a loop tree based on loops included in the computer program.

14. (Original) A dynamic compiling system as recited in claim 13, wherein nested loops are represented in the loop tree as child nodes.

15. (Canceled)

16. (Currently Amended) A ~~computer program~~ dynamic compiler embodied on a computer readable medium ~~for loop optimization within a dynamic compiling~~, the dynamic compiler comprising:

a code segment that ~~interprets byte codes~~ receives a byte code of a computer program ~~having a loop structure, wherein the loop structure includes a loop exit test to be performed during each loop iteration and counting a number of times each of the byte codes is executed~~;

a code segment that determines whether native code is available that corresponds to a subroutine of the computer program including the byte code, and executes the native code when the native code is available;

a code segment that increments a counter when the native code is not available, the counter tracking a number of times the byte code is interpreted;

a code segment that interprets the byte code when the counter is less than a threshold value; and

a code segment that compiles ~~the loop structure during the execution of the computer program; and~~ the subroutine containing the byte code when the counter is greater than the threshold value and native code is not available

~~a code segment that creates an unrolled loop structure during the compiling operation, the creating of the unrolled loop structure being initiated by the byte codes that are often used, as identified by the number of times each of the byte codes is executed, and wherein the unrolled loop structure includes a plurality of loop bodies based on the loop structure~~.

17. (Currently Amended) ~~A computer program as recited in~~ The dynamic compiler of claim 16, wherein the subroutine has a loop structure, wherein the loop structure includes a loop exit test to be performed during each loop iteration, the

compiling comprising creating an unrolled loop structure, the unrolled loop structure includes a plurality of loop bodies based on the loop structure ~~the unrolled loop structure includes the loop exit test, and wherein the loop exit test is performed once for each iteration of the plurality of loop bodies~~.

18. (Currently Amended) ~~A computer program as recited in~~ The dynamic compiler of claim 17, wherein the code segment that compiles further ~~comprising~~ comprises a code segment that performs loop clean up.

19. (Currently amended) ~~A computer program as recited in~~ The dynamic compiler of claim 18, wherein the loop clean-up includes optimizing multiple fall-in loop structures.

20. (Currently amended) ~~A computer program as recited in~~ The dynamic compiler of claim 19, wherein the loop clean-up includes optimizing nested loop structures having invariant operations.

21. (New) The method of claim 1 wherein the byte code is a backward-branching byte code.

22. (New) The method of claim 1 wherein the byte code is a subroutine call.